Introduction to SQL

Class 9

Course Overview

- Introduction to SQL
  - Databases, Tables
  - Classification of SQL – DDL, DML, DCL, TCL
    - DDL – CREATE, ALTER, DROP
    - DML – SELECT, INSERT, UPDATE, DELETE
    - DCL – GRANT, REVOKE
    - TCL – COMMIT, ROLLBACK, SAVEPOINT
  - Data types, Operators
  - Keys – Primary, Foreign, Composite, Unique, Alternate
  - Integrity Constraints – Domain Integrity Constrains, Entity Integrity Constraints, Referential Integrity Constraints
  - Joins – Outer Joins, Left Outer Joins, Right Outer Joins, Inner Joins.
  - Queries, Subqueries, Functions, Flow Control (IF, CASE, WHILE, FOR, LOOP), Stored Procedures,Stored functions
  - Views
  - Indexes, Cursors, Triggers, Events
  - Concurrency and locking (Implicit locks, explicit locks, row level locks, table level locks, database level locks)
  - Tuning SQL queries and optimizing performance
  - SQL Databases vs NoSQL Databases
  - ACID, CAP
  - How SQL databases internally works

CONTROL FLOW Functions(IF)

- SELECT IF(condition, if_condition_true, if_condition_false)
  - ***Condition*** is a condition specified in the IF function
  - ***If_condition_true*** is the value returned by the function
  - ***If_condition_false*** is the value returned by the function
- select if('nisarg'='nirali','True','False') as 'Simple IF statement';
- select if('nisarg'='nirali','True','False') as 'Simple IF statement';

```
Select studentname,
IF(classroom IS NULL, 'Not Applicable',classroom) as 'Classroom',
IF(grade IS NULL, 'Not Applicable',grade)as 'Grade'
from tblstudent;
```

CONTROL FLOW Functions(IF)

```
select studentname , IF(Grade='A+','Brilliant Student','Average Student') as 'Student Grade' from tblstudent;
```

Use IF statement with the COUNT function

```
SELECT
   COUNT(IF(Grade='A+',1,NULL)) 'Brilliant Student',
   COUNT(IF(Grade = 'B+', 1, NULL)) 'Average Student',
   COUNT(IF(Grade = 'D-', 1, NULL)) 'Weak Student'
FROM tblstudent;
```

CONTROL FLOW Statement (CASE)

- A cleaner way of writing multiple IF conditions
- The CASE expression goes through conditions and returns a value when the first condition is met
- Once a condition is true, it will stop reading and return the result.
- If no conditions are true, it returns the value in the ELSE clause.
- If there is no ELSE part and no conditions are true, it returns NULL.
- There are two types of case expression
  - Simple CASE expression
    - The simple CASE expression matches the input with the values for equality and returns the corresponding result.
  - Searched CASE expression
    - The searched case expression evaluates the expression specified in the WHEN clause and returns the corresponding value. The syntax is the following:

```
Simple CASE expression
Select
Case expression
when static_value_1 then output_1
when static_value_2 then output_2
..
Else else_output
End
```

CONTROL FLOW Statement (SIMPLE CASE)

```
select studentname, grade,
case grade
when 'A+' then 'Intelligent'
when 'B+' then 'Average Student'
when 'D' or 'D-' then 'Weak student' end as 'grade description'
from tblstudent;
```

CONTROL FLOW Statement (SIMPLE CASE)

```
select studentname, grade,
case grade
when 'A+' then 'Intelligent'
when 'B+' then 'Average Student'
when 'D' or 'D-' then 'Weak student' end as 'grade description'
from tblstudent where grade is not null
order by
case grade
when 'A+' then 'Intelligent'
when 'B+' then 'Average Student'
when 'D' or 'D-' then 'Weak student' end   asc
```

CONTROL FLOW Statement (SIMPLE CASE)

```
SELECT CustomerName, City, Country
FROM Customers
ORDER BY
(CASE
    WHEN City IS NULL THEN Country
    ELSE City
END);
```

The following SQL will order the customers by City. However, if City is NULL, then order by Country:

CONTROL FLOW Function (SEARCHED CASE)

```
Select
Case
when expression_1 then output_1
when expression_2 then output_2
..
Else else_expression_output
End
```

CONTROL FLOW Function (SEARCHED CASE)

```
select studentname,
case
when studentname like '%Chauhan%' Then 'Hasmukh Chauhan'
when studentname like '%Bhatt%' then 'Jivan Bhatt'
when studentname like '%Upadhyay%' then 'Lalshankar Upadhyay' end
as ParentName,
classroom,grade
from tblstudent
where classroom is not null;
```

CONTROL FLOW Function (SEARCHED CASE)

```
select
Studentname,
    case when count(subject)=0 then 'Details Unavailable' else count(sub
ject) end as 'SubjectsTaken'
from tblstudent a left join tblMarks b on a.ID=b.student_id
group by student_id, studentname
order by b.student_id desc;
```

Stored Functions

- Stored functions in MySQL are used to encapsulate a set of SQL statements into a single, reusable routine (function). They can be used in a variety of situations, including:
  - To perform complex calculations or data manipulations that need to be reused throughout the application.
  - To encapsulate business logic or validation rules that need to be applied consistently across multiple parts of the application.
  - To simplify the application's SQL code by abstracting away complex or repetitive queries.
  - To improve performance by allowing the database to optimize the execution of the stored function.
  - To improve security by allowing the database to execute certain privileged operations without giving the application direct access to the underlying tables.
  - To make the database more modular by breaking it down into smaller, reusable components.

Stored Functions

- Characteristics
  - Stored functions return a single value
  - CREATE ROUTINE database privilege is needed to create a stored function

```
DELIMITER $$

CREATE FUNCTION fun_name(fun_parameter(s))
RETURNS datatype
[NOT] {Characteristics}
fun_body;
```

Returns all the routines from travel database

```
SELECT * FROM information_schema.routines
WHERE routine_schema = 'travel';
```

Stored Functions

- A stored function is a user-defined function that can be used in SQL statements.
- A stored function is defined using the CREATE FUNCTION statement and can take one or more input parameters and return a single value or no value.
- Stored functions can contain SQL statements, control flow statements, and variable assignments.
- Stored functions are similar to stored procedures, but they return a value whereas procedures do not.
- A stored function can be called from a SELECT statement, a SET statement, or from within another stored function or stored procedure.
- Stored functions can be used to encapsulate complex business logic or data validation rules.
- Stored functions are precompiled, which means that the database server performs some optimization on the function when it is created, making its execution faster.
- Stored functions can be used in various contexts, including SELECT, UPDATE, DELETE and INSERT statements.
- Stored functions can be recursive, meaning that a function can call itself, to a certain level.
- Stored functions can be called from triggers, events and views.

Stored Functions

Parameter Name	Descriptions
fun_name	It is the name of the stored function that we want to create in a database. It should not be the same as the built-in function name of MySQL.
fun_parameter	It contains the list of parameters used by the function body. It does not allow to specify IN, OUT, INOUT parameters.
datatype	It is a data type of return value of the function. It should any valid MySQL data type.
characteristics	The CREATE FUNCTION statement only accepted when the characteristics (DETERMINISTIC, NO SQL, or READS SQL DATA) are defined in the declaration.
fun_body	This parameter has a set of SQL statements to perform the operations. It requires at least one RETURN statement. When the return statement is executed, the function will be terminated automatically. The function body is given below: BEGIN -- SQL statements END $$ DELIMITER

Stored Functions - Examples

A function to calculate the total cost of a purchase order:

```
CREATE FUNCTION get_total_cost(order_id INT)
RETURNS DECIMAL(10,2)
BEGIN
  DECLARE total_cost DECIMAL(10,2);
  SELECT SUM(price * quantity) INTO total_cost
  FROM order_items
  WHERE order_id = order_id;
  RETURN total_cost;
END;
```

Stored Functions - Examples

A function to check if an email address is valid:

```
CREATE FUNCTION is_valid_email(email VARCHAR(255))
RETURNS BOOLEAN
BEGIN
  DECLARE email_regex VARCHAR(255) DEFAULT '^[a-zA-Z0-9._%+-
]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$';
  RETURN email REGEXP email_regex;
END;
```

Stored Functions - Examples

A function to calculate the average salary of employees in a department:

```
CREATE FUNCTION get_avg_salary(department_id INT)
RETURNS DECIMAL(10,2)
BEGIN
  DECLARE avg_salary DECIMAL(10,2);
  SELECT AVG(salary) INTO avg_salary
  FROM employees
  WHERE department_id = department_id;
  RETURN avg_salary;
END;
```

Stored Functions - Examples

A function to check if a given date is a weekend or not:

```
CREATE FUNCTION is_weekend(date DATE)
RETURNS BOOLEAN
BEGIN
  RETURN (WEEKDAY(date) = 5 OR WEEKDAY(date) = 6);
END;
```

Stored Functions - Examples

A function to calculate the total number of orders for a customer

```
CREATE FUNCTION get_total_orders(customer_id INT)
RETURNS INT
BEGIN
   RETURN (SELECT COUNT(*) FROM orders WHERE customer_id = customer_id);
END;
```

Stored Functions - Examples

A function to calculate the total number of products in a category:

```
CREATE FUNCTION get_total_products(category_id INT)
RETURNS INT
BEGIN
   RETURN (SELECT COUNT(*) FROM products WHERE category_id = category_id);
END;
```

Stored Functions - Examples

A function to check if a given string is a palindrome or not:

```
CREATE FUNCTION is_palindrome(string VARCHAR(255))
RETURNS BOOLEAN
BEGIN
  RETURN (string = REVERSE(string));
END;
```

Stored Functions - Examples

A function to calculate the total cost of shipping for an order:

```
CREATE FUNCTION get_shipping_cost(order_id INT)
RETURNS DECIMAL(10,2)
BEGIN
   RETURN (SELECT SUM(cost) FROM shipping WHERE order_id = order_id);
END;
```

Stored Functions - Examples

A function to convert lbs to kg

```
DELIMITER //
CREATE FUNCTION lbs_to_kg(lbs MEDIUMINT UNSIGNED)
RETURNS MEDIUMINT UNSIGNED
DETERMINISTIC
BEGIN
  RETURN (lbs * 0.45359237);
END//
DELIMITER ;

SELECT a.plane, max_weight AS max_lbs,
  lbs_to_kg(max_weight) AS max_kg
FROM airplanes a INNER JOIN manufacturers m
  ON a.manufacturer_id = m.manufacturer_id
WHERE m.manufacturer = 'airbus'
ORDER BY a.plane;
```

Stored Functions

- `DROP FUNCTION IF EXISTS lbs_to_kg;`

Connect with me on LinkedIn

Please subscribe to our YouTube channel

Check out my GitHub profile

Follow me on Twitter(X)

Thank you